Datavetenskap

Patrik Isaksson & Mikael Lindmark

Skapandet av ett online-spel

Examensarbete, C-nivå

2004:19

Skapandet av ett online-spel

Patrik Isaksson & Mikael Lindmark

Denna rapport är skriven som en del av det arbete som krävs för att erhålla en kandidatexamen i datavetenskap. Allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Patrik Isaksson

Mikael Lindmark

Godkänd 2004-06-03

Handledare: Thijs Holleboom

Examinator: Stefan Alfredsson

Sammanfattning

Webbaserade Online-spel är idag en växande spelmarknad. Ett flertal spel har de senaste åren växt till flera hundra tusen användare. För att möjliggöra ett sådant spel behövs en databas, ett skriptspråk och eventuellt en spelmotor. Vi har skapat ett spel där användare via en webbläsare kontrollerar och bygger en stad. Spelet går ut på att få den största och mest välmående staden. Det användarna ser i webbläsaren är bara en av de delar som tillsammans interagerar och utgör spelet. Som skriptspråk använder vi oss utav PHP, som med sin skriptmotor genererar den text och bild användarna ser. PHP-koden sköter också de handlingar som användarna gör. All spelinformation om städerna finns lagrad i en databas som uppdateras av PHP-koden. Databasen är av typen MySQL. Det finns även en spelmotor, skriven i C++, som körs hela tiden och vid specifika tidpunkter uppdaterar spelinformationen i databasen. I PHP finns det bra inbyggt stöd för att jobba mot en MySQL-databas, medan för C++ finns ett särskilt API, MySQL++, som kan användas vid MySQL-databasaccesser. För att användaren ska känna igen sig, känna sig trygg och lätt kunna navigera runt på sidan har vi försökt att få alla sidor på spelet att ha samma grundstruktur och utseende.

Den här rapporten beskriver utvecklingen och utmaningarna vid skapandet av ett online-spel. Spelet som vi utvecklat, döpt till BigCity, är vi relativt nöjda med, det följer i stort sett den specifikation, se bilaga B, som vi satte upp i början av examensarbetet. Alla de större delar vi ville ha med förutom handelsavtal har blivit implementerade. Spelet känns dock inte helt färdigt, det skulle nog behöva mer funktionalitet. Det kan ibland kännas som om det inte finns så mycket att göra på spelet, men det har blivit en mycket bra grund att bygga vidare på.

The development of an online-game

Abstract

Web based online games increases today on the game market. A number of games have the last years expanded to several hundred thousands of users. To render possibility for this kind of games, a database, a script language and possibly a game engine. We have created a game where the users control and rule a city with a web browser. The goal of the game is to control the biggest and most healthy city. The view that the user sees in the web browser is just one of the parts that together interacts and constitutes the game. For script language, PHP is used. PHP which script engine generates the text and the images the user sees. The PHP-code also manages the actions the user performs. All game information about the cities is stored in a database, which is updated by the PHP-code. The database is a MySQL-database. There is also a game engine written in C++. The engine is constantly running and updates the game information at specific points of time. PHP has good built-in support for MySQL access, while in C++ there is a special API, MySQL++, which can be used to access the MySQL-database. All the pages in the game have the same base structure and look, so that the users will feel safe and easily can navigate through the game.

This report describes the development of an online game called BigCity. We are relatively satisfied with the game. To a great extent, the game follows the specification, see appendix B, which was composed in the beginning of the bachelor project. All the major parts we wanted to implement, except the trade agreement-part have been implemented. However, the game does not feel quite finished, some more functionality would probably be necessary. Sometimes it feels like there isn't so much to do in the game, but it has become a very good base to continue development on.

Innehållsförteckning

Figurförteckning

Tabellförteckning

1 Inledning

I många tusen år har människor spelat spel av olika slag mot varandra. Detta har även fortsatt in i vår tid och när det första datorspelet, Space War, kom 1962 var det skapat för att två personer skulle spela mot varandra. Sedan dess har många fler dataspel skapats och åtskilliga av dem är skapade för att flera användare samtidigt kan spela dem mot varandra. När World Wide Web (www) blev en del av Internet, tillkom en ny typ av spel inom spelbranschen. Dessa spel bygger på att användare interagerar och spelar med eller mot varandra via sin webbläsare. I denna kategori spel är interaktion och kommunikationen mellan spelarna en av de primära delarna. Ibland är det till och med så att kommunikationen mellan spelarna blir viktigare än själva spelet. I och med utbredningen av bredband har antalet människor som spelar online-spel ökat dramatiskt. Även antalet online-spel av denna typ har ökat. För att utveckla dessa spel krävs det kunskaper inom ett flertal områden. Kunskap om såväl databashantering, programmering och webblayout behövs för att få en produkt av kvalitet som tilltalar användarna.

I denna uppsats beskriver vi spelet vi skapat och de delar spelet är uppbyggt av. Dessa delar är användargränssnitt, spelmotor och databas. Användargränssnittet utgörs av HTML som genererats av php-filerna. Spelmotorn är skriven i C++ och uppdaterar speldatan vid specifika tidpunkter. I databasen finns den speldata som hela spelet är baserat på.

2 Bakgrund och syfte

Ett online-spel är ett spel där användare interagerar med varandra via Internet. Dessa spel finns i många olika varianter och inriktningar. Det område som detta examensarbete inriktar sig mot är den sort som spelas med en webbläsare, till skillnad från andra online-spel, där det krävs speciella klientprogram. Exempel på klientbaserade spel är Counter-Strike och Diablo där användaren styr en person i en grafisk värld i en kamp mot eller med andra användare. Speltypen som spelas med en webbläsare är i grunden enbart baserad på text och bilder. I övrigt saknar de den grafiska spelrepresentation som ofta finns i de klientbaserade spelen.

Informationen som ska visas för användaren i de textbaserade online-spelen finns lagrad i en databas. Användaren ser informationen i HTML-sidor i sin webbläsare. Dessa sidor är genererade av något skriptspråk som hämtar informationen från databasen. Exempel på vanligt använda skriptspråk är ASP, PHP, JSP samt CGI.

De webbaserade online-spelen på dagens marknad finns inom ett fåtal genrer t.ex. managerspel, erövringsspel och fightingspel. Managerspel går ut på att användaren får leda ett lag inom någon sport till framgång. Användaren får coacha laget, styra ekonomin, bygga ut arenan samt köpa eller sälja spelare. Erövringsspel går ut på att vara ledare för ett rike där användaren bygger upp arméer och attackerar andra användare för att kunna utvidga och utveckla sitt rike. Miljön som erövringsspelen utspelar sig i är ofta fantasy- eller science-fiction-inspirerad. Fightinggenren är något ovanliggare är de andra genrerna. I denna genre har användaren en karaktär som ska tränas för att kunna slåss mot olika sorters monster eller andra karaktärer. Karaktären utrustas med olika sorters utrustning och vapen. I Tabell 2-1 finns det några exempel på online-spel inom dessa genrer.

Managerspel	**Erövringsspel**	**Fightingspel**
Hattrick[3]	Utopia[7]	Warriors2[10]
Stolpskott[5]	Orkfia[8]	Virtual online Wrestling[11]
Managerzone[6]	Dominion[9]	

Tabell 2-1 Exempel på online-spel.

Fotbollsspelet Hattrick[3], har ökat från c:a 15 000 användare för fyra år sen till dagens 250 000 användare. Detta visar på det väldiga intresse som finns för online-spel idag.

Detta examensarbete görs för att beskriva skapandet av ett webbaserat online-spel, och en del svårigheter och problem som kan uppstå under utvecklingen.

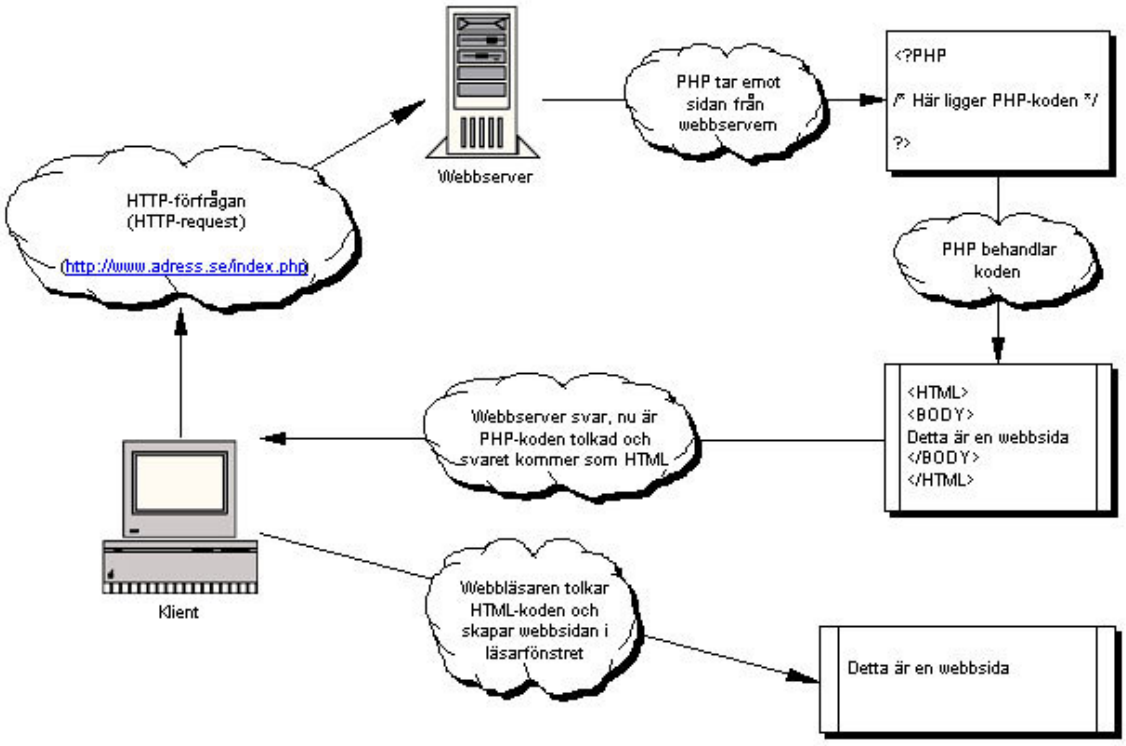
3 Förutsättningar och krav

Förutsättningar för att detta examensarbete ska kunna genomföras är att det finns tillgång till en serverdator med uppkoppling mot Internet. För att lagra information om spelet och spelarna behövs en databas. Serverdatorn behöver även en skriptmotor som har stöd för databashantering. Denna skriptmotor behöver jobba med en webbserver som kan förmedla informationen till spelarna som html-sidor.

Innan examensarbetet började hade vi kunskaper inom C++ och databashantering. PHP hade vi inte kommit i kontakt med så mycket tidigare. Det visade sig dock snart att detta ej skulle sätta några käppar i hjulet för oss. Vi hade tidigare aldrig jobbat mot en databas med C++, detta visade sig inte heller bli några problem, då ett bra API (MySQL++) fanns att tillgå för detta ändamål.

4 Konstruktionslösning

Spelet är i huvudsak uppdelat i tre stora delar, ett användargränssnitt, en spelmotor och en databas som dessa två arbetar mot. I Figur 4-1 visas det hur klienten begär en PHP-sida. Webbservern tar emot begäran och skickar vidare den till PHP-skriptmotorn som tolkar koden till en HTML-sida. Webbservern får tillbaka HTML-sidan och skickar vidare den till klienten.

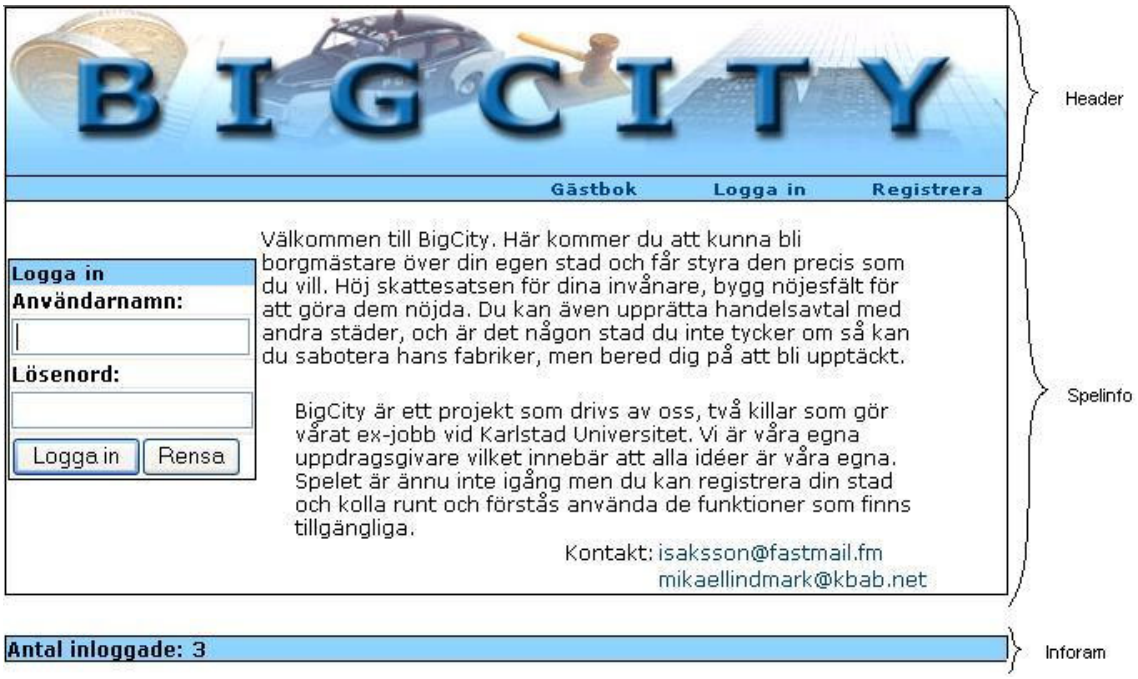


Figur 4-1 Sambandet mellan webbläsare, webbserver och PHP

4.1 Användargränssnitt

Användargränssnittet är uppbyggt av PHP och HTML. PHP-motorn genererar HTML-kod, som sedan visas för användaren, av PHP-kod. Användaren märker alltså aldrig av PHP-koden utan ser endast HTML-tolkningen av koden.

Användargränssnittet i spelet är uppbyggt så att alla sidor har samma ”look-and-feel”. För att åstadkomma detta används en header vilket gör att det blir samma grundstruktur på alla sidor. I denna header finns spelets logotyp och en spelmeny. Header-filen inkluderas sedan högst upp av alla filer som används när användaren är inloggad.

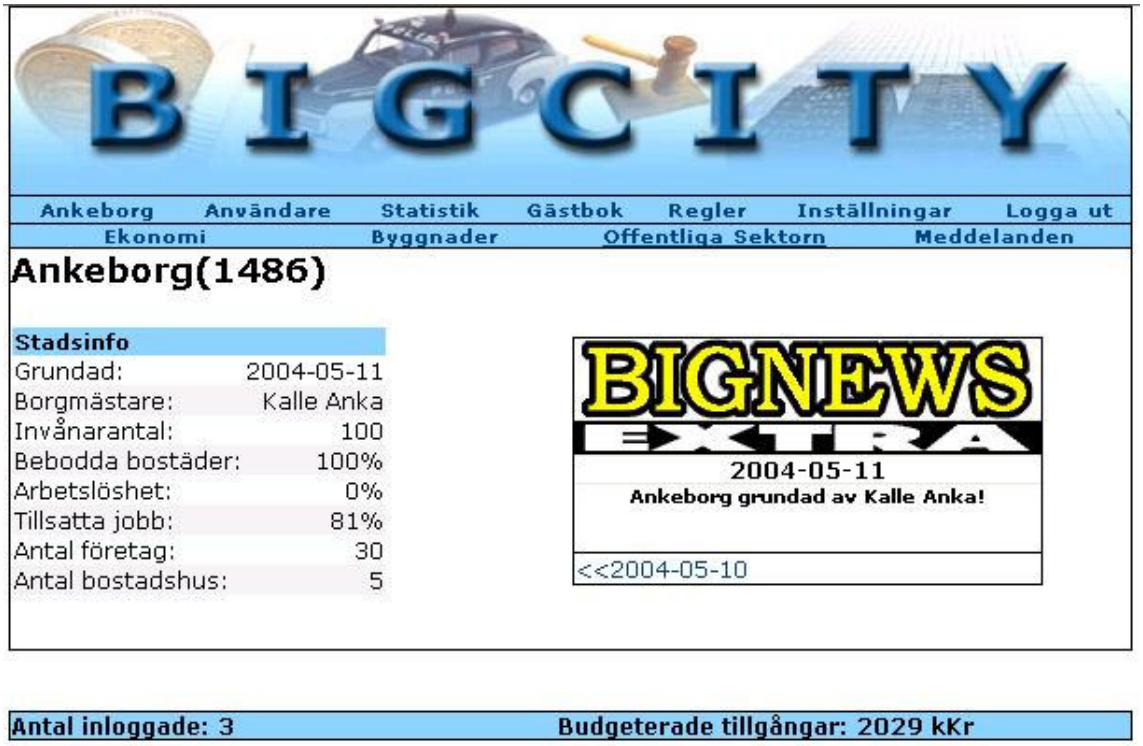


Figur 4-2 Inloggningssidan

Huvudfönstret är uppdelat i två ramar. En huvudram där användaren kan se och förändra innehållet. Denna sida förändras när användaren navigerar runt inom spelet. Den andra ramen är en liten snabbinformationsram där det står hur mycket tillgångar användaren har att tillgå och antalet användare som är inloggade i spelet just för tillfället. I Figur 4-2 visas sidan där användaren loggar in på spelet.

4.1.1 Inloggning

När användaren kommer till spelet laddas login.php i den stora ramen i index.php. Det är här som användaren får skriva in användarnamn och lösenord i ett formulär. I menyn finns det länkar till registrering och till en gästbok. När användaren trycker på knappen laddas sidan om med användarnamn och lösenord som inparametrar. Var användarnamn och lösenord korrekta kommer användaren till sidan för sin stad (Figur 4-3) eller så skrivs ett felmeddelande ut om uppgifterna inte stämde.



Figur 4-3 Användarens stad

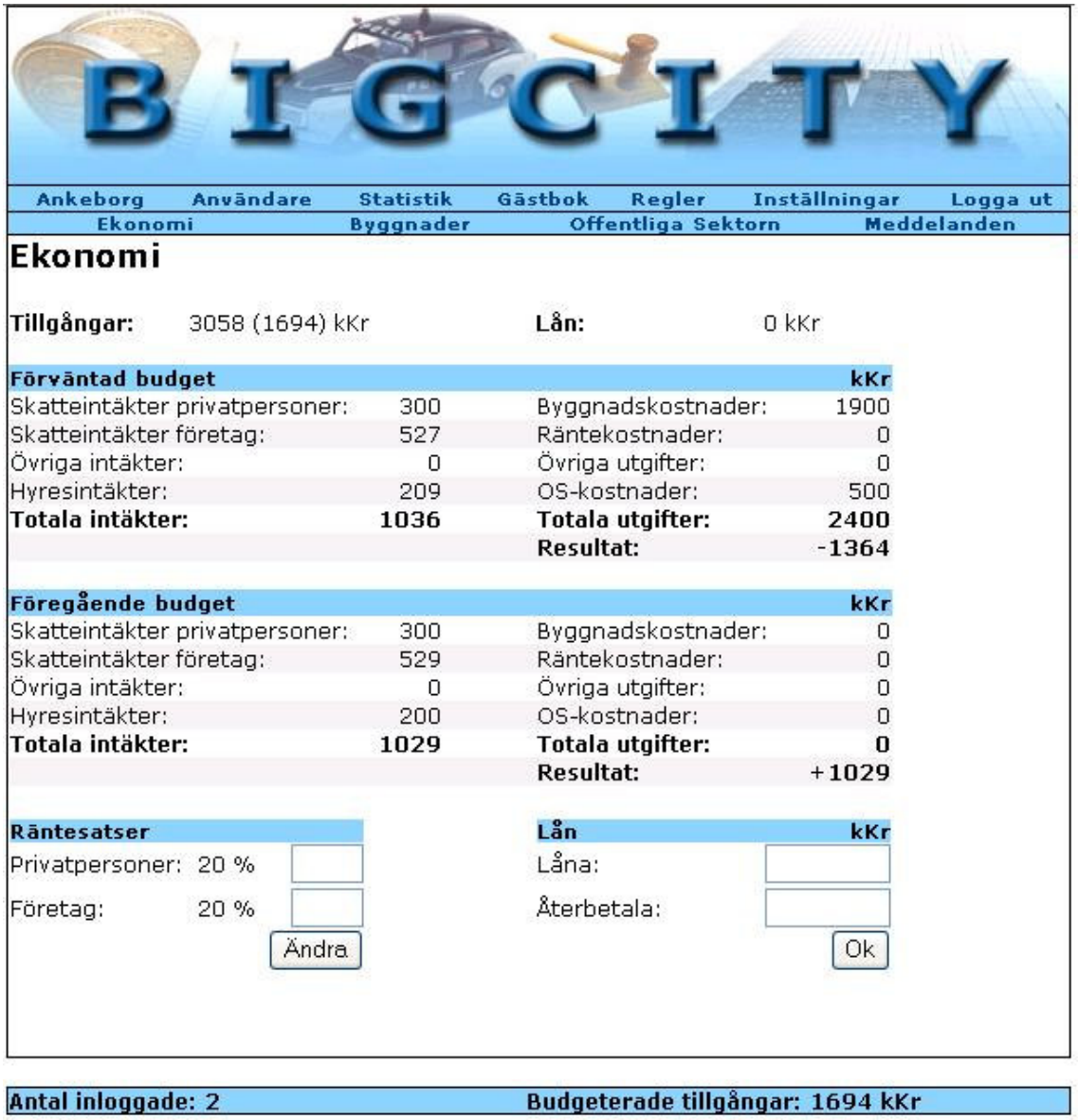
4.1.2 Användarens stad

När användaren loggat in öppnas en sida som visar en översikt över användarens stad. Högst upp under logotypen finns det en meny. Det är denna meny som används när användaren navigerar runt på sidan. Informationen som visas om staden är när den grundades, borgmästare, invånarantal bebodda bostäder, arbetslöshet, tillsatta jobb, antal företag och antal bostadshus. På denna sida visas även stadens tidning, Bignews. Här finns det nyheter om vad som hänt i staden. Användaren kan bläddra sig bakåt och titta på tidningar från den senaste veckan. I användarens egen stad finns det fyra underrubriker: ekonomi, byggnader, offentliga sektorn och meddelande.

4.1.3 Stadens ekonomi

I Figur 4-4 visas en översikt över användarstadens ekonomi. På denna sida visas och sköts det mesta som har med ekonomi att göra så som tillgångar, intäkter och utgifter. Även skattesatser och lån kontrolleras härifrån. Användaren ser nuvarande och föregående budget. I budgeten syns skatteintäkter för privatpersoner, skatteintäkter för företag, övriga intäkter,

hyresintäkter, totala intäkter, byggnadskostnader, räntekostnader, övriga utgifter, offentliga sektorkostnader, totala utgifter och resultat.

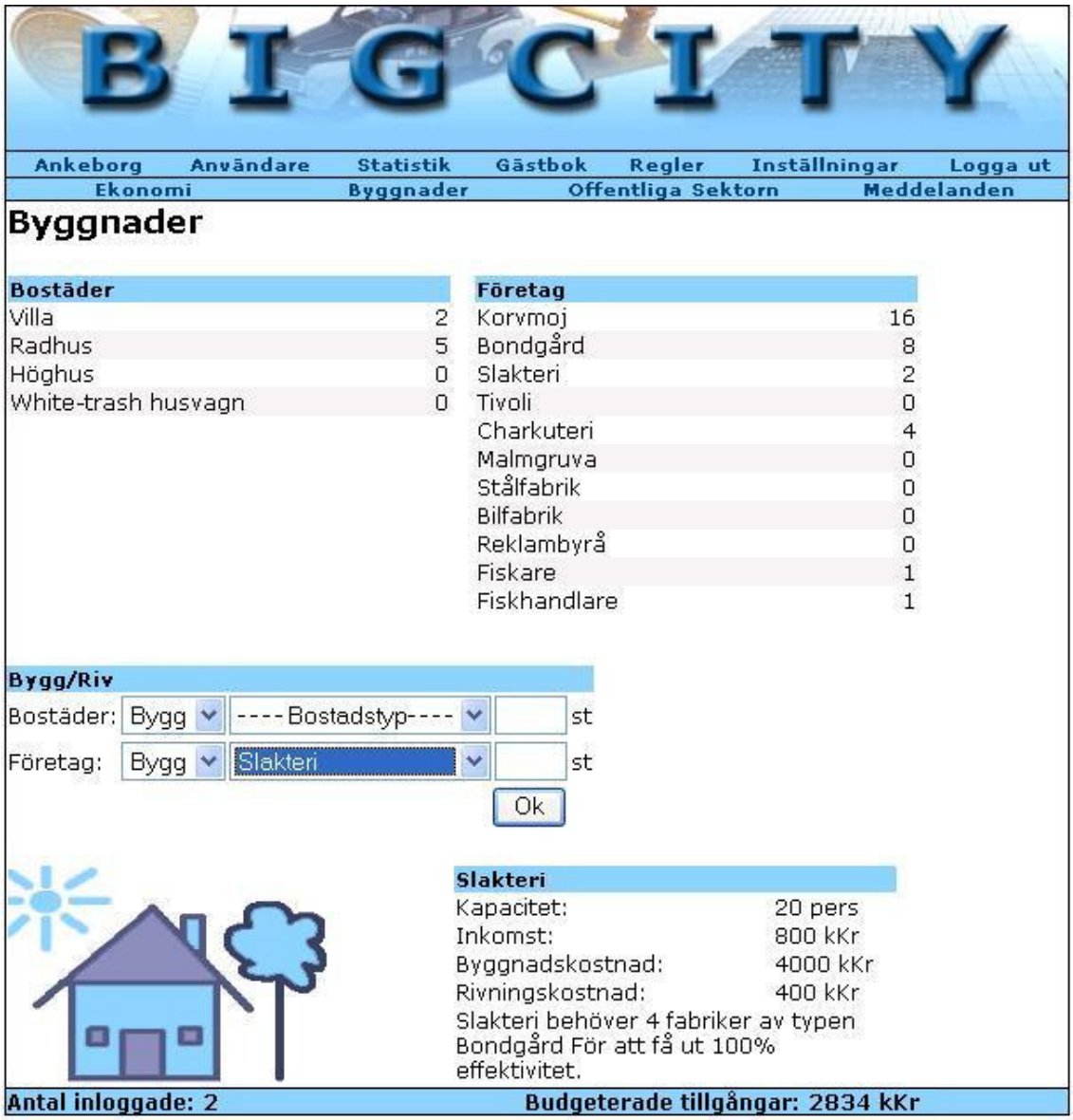


Figur 4-4 Ekonomisidan

4.1.4 Byggnader

Varje stad har ett antal byggnader, och det är förstås här som dessa administreras. I Figur 4-5 visas en översikt av alla byggnadstyper och antalet som finns av varje i staden. Byggnadstyperna är uppdelade i bostäder och företagsbyggnader. Längre ner på sidan finns ett formulär för att bygga nya och riva gamla byggnader. I en drop-down-meny kan

användaren välja vilken typ av byggnad att bygga. När byggnad är vald kommer det upp information om byggnaden i en liten ram. Informationen som visas för företagsbyggnader är: kapacitet, inkomst, byggnadskostnad, rivningskostnad och om företaget är beroende av några andra företag. För bostäder visas: kapacitet, hyra, byggnadskostnad och rivningskostnad.



Figur 4-5 Byggnader

4.1.5 Offentliga sektorn

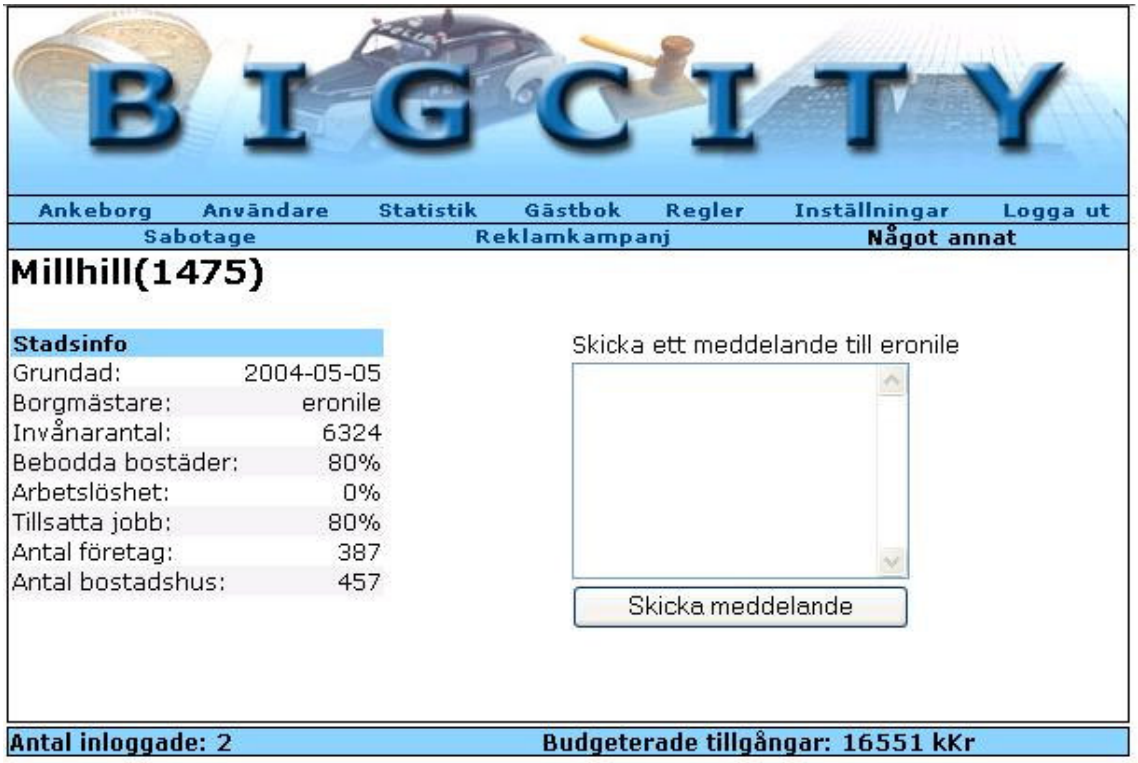
Den offentliga sektorn består av fem avdelningar: infrastruktur, polis, sjukvård, kultur och lokaltrafik. I Figur 4-6 visas det hur information om de olika avdelningarna presenteras för användaren. Antal representerar hur många människor i staden som avdelningen räcker till.

Täckning är antal/population. Health-level är en representation av hur bra skick avdelningen är i. För att inte Health-leveln ska sjunka måste användaren avsätta pengar så avdelningarna har resurser att röra sig med. För att förändra en avdelning finns det ett formulär där användaren väljer om avdelningen ska utökas eller minskas, vilken avdelning operationen ska utföras på samt hur mycket avdelningen ska utökas/minskas.

Figur 4-6 Offentliga sektorn

4.1.6 Andra användares städer

När användaren tittar på en annan användares stad, är det i stort sett samma information som visas som när användaren tittar på sin egen stad. Det som skiljer är att användaren inte ser den andra stadens tidning. Anledningen till att det är på detta vis är att användarna inte ska veta allt som sker i andra städer. I Figur 4-7 visas hur informationen presenteras och att det även finns ett formulär för att skicka meddelande till användaren.



Figur 4-7 Andra användares städer

Figur 4-8 Sabotage

4.1.7 Sabotage

I Figur 4-8 visas sidan där användaren kan utföra sabotage i andra städer. Till vänster på sidan finns en drop-down-meny där de olika sabotagealternativen finns. När ett alternativ har valts

kommer det upp information om det i en ram till höger på sidan. När användaren sedan begrundat de olika alternativen bekräftas valet med ok-knappen. Sabotaget utförs sedan och användaren får en utskrift som säger om det gick bra eller inte.

4.1.8 Övriga vyer

Övriga vyer har inte någon direkt spelfunktion utan visar mest information, som t.ex. statistiksidan som listar de rikaste och största städerna, därför tar vi inte upp dem närmare här.

4.2 PHP-sidor

Denna del handlar om vad php-sidorna gör bakom användargränssnittet, alltså de beräkningar som utförs för att visa data och när användare utför diverse operationer.

4.2.1 Index.php:

Denna sida är huvudsidan. Index.php delar in fönstret horisontalt i två ramar, en stor huvudram och en mindre inforam. Innehållet i den stora ramen förändras då användaren navigerar runt i spelet. Inforamen visar samma sida hela tiden, nämligen bottom.php.

4.2.2 Login.php:

Denna sida laddas från början i den stora ramen i index.php. Det är här som användaren får skriva in användarnamn och lösenord. På sidan finns det ett formulär för användarnamn och lösenord. I formuläret finns även en knapp. När användaren trycker på knappen laddas sidan om med användarnamn och lösenord som inparametrar. Innan några utskrifter sker kontrolleras det om knappen har blivit intryckt. I nästa steg kontrolleras det om användarnamn eller lösenordet var tomma. Var någon utav dem det skrivs ett felmeddelande ut. Var både användarnamn och lösenord ifyllda så ställs en fråga mot databasen och cityId begärs där användarnamn och lösenordet finns. Om databasen returnerar en rad, skapas en sessionvariabel med användarens cityId som värde. Sessionsvariabeln används för att identifierare användaren under den tid som användaren är inloggad. I databasen uppdateras ett fält som heter lastChanged. Detta fält lagrar tiden för när användaren senast gjorde någonting på sidan. Slutligen körs ett anrop till den inbyggda funktionen header som gör så att

användaren kommer till sin stads sida. Fanns inte kombinationen användarnamn och lösenord i databasen så skrivs ett felmeddelande ut

4.2.3 Viewuser.php:

Här presenteras information om en stad. Sidan genereras lite olika beroende på om användaren tittar på sin egen stad eller någon annans. Anledningen till att den visas annorlunda är att användarna inte ska veta allt om sina motspelares städer. Sidan kan ta in ett id som argument. Skickas det inte med något argument listas alla användare och det går då klicka på dem för att se deras stad. Är däremot id medskickat jämförs detta mot sessionvariabeln. Är de lika är det sin egen stad användaren kollar på. Den information som visas är datum då staden grundades, borgmästare, invånarantal, bebodda bostäder, arbetslöshet, tillsatta jobb, antal företag och antal bostadshus. Det som skiljer sig om det är sin egen eller någon annans stad användaren tittar på är att på användarens egen sida finns en löpsedel från stadens tidning. Är det däremot någon annans stad användare tittar på finns det istället för löpsedeln ett formulär för att skicka ett meddelande till användaren. Tidningslöpsedeln genereras av bignews.php och meddelandeformuläret av message_area.php. Om användaren anropar sidan manuellt med ett id som inte finns i databasen skrivs ett felmeddelande ut.

4.2.4 Bignews.php:

Bignews.php genererar en löpsedel där användaren kan se vad hänt i sin stad. Som default laddas dagens löpsedel, men användaren kan även bläddra och se den senaste veckans löpsedlar. De händelser som visas på löpsedeln har sparats i databasen då de inträffat. Händelser som sparas är t.ex. sabotage från andra städer, rivning av bebodda hus, när populationen når över ett visst antal (10^2, 10^3, …, 10^n) och när användarens stad blivit utsatt för reklamkampanjer.

Det datum som användaren vill kolla nyheterna på skickas med som ett argument till bignews.php. Är det inte något argument medskickat blir det dagens datum. Alla händelser som rör staden och som sparats under det aktuella datumet plockas sedan ut ur databasen. Sedan kontrolleras varje rad som plockats ut ur databasen. Är sabotageId inte null mappas informationen mot sabotagedelen i databasen och skriver ut rätt information.

4.2.5 Economy.php:

Economy.php visar en ekonomisk översikt för användaren. För att hämta ekonominformationen från databasen används funktionen getEconomy som inkluderas via filen functions.php. GetEconomy hämtar och räknar enbart ut den aktuella veckans ekonomiuppgifter, medan föregående veckas uppgifter finns lagrade direkt i databasen och således kan hämtas utan några uträkningar. Det som returneras från getEconomy är en associativ array, alltså en array där positionerna kan accessas via namn. Ekonomiarrayen ser ut såhär: $economy[money, peopleTaxIncome, corpTaxIncome, otherIncome, rentIncome, totalIncome, buildingCost, interestCost, otherExpenses, publicSectorCost, totalExpenses, expectedMoney, peopleTax, corpTax, loan].

4.2.6 Build.php

Build.php har hand om användarens byggnader. Här visas en översikt av de byggnader som finns i staden och även ett formulär där användaren kan riva eller bygga nya byggnader.

I drop-down-menyer kombinerat med ett textfält kan användaren välja att bygga eller riva, vilken typ av byggnad operationen ska utföras på och antal byggnader som berörs. Vid förändringar i drop-down-menyn över byggnadstyper laddas en frame med information över just den byggnadstyp användaren valt.

Om användaren har tryckt på knappen för att bygga/riva kontrolleras först att det är giltig data som användaren har matat in. Med giltig data menas att en byggnadstyp är vald och att ett positivt tal står i antal att bygga/riva. Om användare valt att riva byggnader kontrolleras det att det finns minst så många byggnader av typen som ska rivas. Om täckning finns för rivning kontrolleras det om folket i staden har tillräckligt med byggnader att bo i efter rivningen. Finns det inte det flyttar överskottet ifrån staden och en tidningsrubrik genereras. Går allt bra sätts byggnadskostnaden till aktuell rivningskostnad som är en tiondel av uppföringskostnaden. Valde användaren att bygga, sätts byggnadskostnaden till aktuell uppföringskostnad. Sedan kontrolleras det om det finns tillräckligt med pengar i de budgeterade tillgångarna. Finns det inte det skrivs ett felmeddelande ut och inga byggnader byggs.

Har allt gått bra så här långt ställs en fråga mot databasen där det nya antalet byggnader sätts in. Har användaren rivit alla byggnader av en typ så tas posten helt bort ifrån byggnadsrelationen.

4.2.7 Public_sector.php

Denna fil har hand om stadens offentliga sektor. Den offentliga sektorn består av fem avdelningar: infrastruktur, polis, sjukvård, kultur och lokaltrafik. Användaren får upp en översikt av den offentliga sektorn där det står hur många personer avdelningens tjänster räcker till, hur stor procent den täcker, dess health-level och hur mycket resurser som är avsatta till avdelningen. Användaren kan i olika formulär utöka/minska en avdelnings täckning eller avsätta pengar till en avdelning.

Om användaren valt att utöka/minska kontrolleras det först att det är giltig data som sänts, alltså enbart positivt heltal i inmatningsfältet och en avdelning vald. Nästa sak som kontrolleras är om det finns tillräckligt med pengar för operationen. Har användaren valt att minska avdelningens utsträckning kontrolleras det om det finns tillräckligt med enheter för det. Har allt gått bra nu så ställs det en fråga mot databasen där datan om användarens offentliga sektor uppdateras.

Har användaren tryckt på knappen för att avsätta pengar till en avdelning kontrolleras det även här att det är giltig data. Om datan var korrekt uppdateras resurserna för avdelningen i databasen.

4.2.8 Header.php

Header.php inkluderas av alla filer som används när användaren är inloggad. För att användaren inte ska kunna komma till en sida där inloggning krävs, så kontrollerar denna fil om det finns någon sessionvariabel som är kopplad till användarens webbläsare. Finns det inte någon sessionvariabel laddas en sida där det står att användaren måste vara inloggad för att se sidan som först försökte laddas.

Header.php kontrollerar också när användaren gjorde någonting på sidan senast. Var det längre än tio minuter sedan så kastas användaren ut. Detta för att förhindra att användare glömmer logga ut på t.ex. offentliga datorer och någon förstör för dem. Var det inte längre än

tio minuter sedan användaren gjorde nåt på sidan så uppdateras det i databasen när användaren senast gjorde något till aktuell tid.

Sedan genereras den html som är högst upp på alla sidor när användaren är inloggad. Överst är det BigCitys logotyp och under den huvudmenyn.

4.2.9 Bild.php

Denna sida tar in en sökväg till en bild som argument och visar sedan bilden. Anledningen till denna lösning är att en del webbläsare, t.ex. Opera skriver ut information om sökvägen till bilden vilket gjorde att det blev rullningslister i ramen där bilden skulle visas.

4.2.10 Bottom.php

Denna sida visas hela tiden i den lilla ramen längst ner på huvudsidan. En fråga ställs mot databasen där de användare som laddat en sida de senaste 10 minuterna plockas ut. Antalet av dessa skrivs då ut som antal inloggade. Är användaren dessutom själv inloggad hämtas stadens ekonomi med hjälp av getEconomy-funktionen och budgeterade tillgångar skrivs ut.

4.2.11 Change.php

Change.php har hand om användarens inställningar. Användaren kan på denna sida ändra användarnamn, e-mail, namn och lösenord. För att ändra inställningar måste användaren ange det gamla lösenordet korrekt. Ett nytt lösenord måste skrivas in två gånger för att förhindra att användaren skriver fel. Det gamla lösenordet är det första som kontrolleras när användaren tryckt på ändra-knappen. Sedan kontrolleras det om användaren skrivit i något nytt lösenord, och om användaren gjort det kontrolleras det att det är samma nya lösenord i båda fälten. Vidare så kontrolleras det att användarnamn, e-mail och riktigt namn inte är tomt. Gick allt bra ställs en uppdateringsfråga mot databasen och användaren blir informerade om det. Gick något snett skrivs ett felmeddelande ut om det.

4.2.12 Constants.php

Här finns de konstanter som används i spelet.

4.2.13 Create.php

När en användare skapar ett konto så är det i create.php som det sker. Användare får fylla i användarnamn, lösenord, e-mail, riktigt namn och stadsnamn i ett formulär. Först kontrolleras att inget fält är tomt. Sedan kontrolleras att ingen redan registrerad användare har samma användarnamn eller e-mail. Gick detta bra skapas ett användarkonto med en stad, grundbyggnader, offentlig sektor och ekonomi och en text skrivs ut att kontot är skapat. Lyckades inte registrering skrivs detta ut och användaren får försöka igen.

4.2.14 Destroy.php

Denna fil är en sorts admin-fil där alla användare listas med en länk för att ta bort användarens konto. Här läggs även testscript in vid behov.

4.2.15 Footer.php

Footer.php inkluderas av alla filer och avslutar html-dokumentet och stänger uppkopplingen mot databasen.

4.2.16 Functions.php

I Functions.php finns det funktioner som räknar ut en stads ekonomiska översikt. De filer som behöver detta inkluderar functions.php och kan sedan anropa funktionen getEconomy. getEconomy tar emot ett användar-id som argument, hämtar ut användarens uppgifter i ekonomirelationen och gör sedan beräkningar och returnerar en associativ array med datan.

4.2.17 Header_notlogged.php

Header_notlogged.php är en variant av header.php som inte kontrollerar om användaren är inloggad. Denna fil används alltså på de flesta filer som visas när användaren inte har loggat in.

4.2.18 Logout.php

Denna fil anropas när användaren loggar ut. Logout.php gör inte mycket mer än att ta bort sessionvariabeln.

4.2.19 Manual_sql.php

Detta är en admin-sida skriven för att manuellt kunna ställa SQL-frågor mot databasen, och den är endast åtkomlig för administratörer. Om SQL-frågan genererar ett resultat visas detta.

4.2.20 Mytown_menu.php

Mytown_menu.php genererar en meny över ekonomi, byggnader och den offentliga sektorn som inkluderas av de filer som rör användarens egen stad.

4.2.21 Notloggedin.php

Denna sida visas när en icke inloggad användare har försökt att nå en fil som kräver att användaren är inloggad.

4.2.22 Notmytown_menu.php

Denna sida inkluderas när användaren kollar på en stad som inte är användarens egen. Sidan genererar en meny med länkar till sabotage & reklam.

4.2.23 Rules.php

I rules.php visas spelets regler. Användaren kan här läsa om hur spelet och dess olika delar fungerar och är uppbyggda.

4.2.24 Sabotage.php

Sabotage.php har hand om de lite mer oärliga inslagen av spelet. Användare kan här fiffla med bokföring, bränna ner företag eller sabotera infrastrukturen i andra städer. I en drop-down-meny kan användaren bläddra mellan de olika alternativen. Vid förändring i menyn laddas en ram med information om aktuell sabotagetyp och dess kostnad. När användaren tryckt på knappen kontrolleras det att en sabotagetyp är vald. Sedan plockas kostnad för sabotagetypen ut och jämförs med användarens budgeterade tillgångar. Om operationen lyckas eller inte bestäms av ett slumptal och polisstyrkan hos städerna. Ju sämre poliskårer i städerna (både den drabbade och i den som saboterar) desto större chans att operationen lyckas. Det finns tre möjliga utgångar av operationen. 1: Operationen lyckas helt och den drabbade staden har ingen aning om vem som ligger bakom. 2: Operationen lyckas delvis.

Attentatet lyckas då, men den drabbade staden har mycket starka misstankar om vem det är som ligger bakom dådet. 3: Operationen misslyckas. Staden som utförde dådet åker fast och får böta en summa pengar.

4.2.25 Sabotage_info.php

Sabotage_info.php tar emot ett sabotage-id som argument och visar information om den sabotagetyp som matchar sabotage-id:t. Är det inte medskickat något sabotage-id visar sidan ingenting.

4.2.26 Stats.php

Denna sida visar listor på vilka städer som är rikast och vilka städer som har flest invånare. Inga krångliga uträkningar sker här utan enbart en fråga mot databasen där städerna plockas ut och sorteras i ordning efter pengar eller invånarantal.

4.2.27 Users.php

Här listas de användare som är inloggade för tillfället. I relationen users finns det ett fält, lastChanged, innehållandes en tidsstämpel som uppdateras varje gång användaren laddar en sida då han är inloggad. En tidsstämpel är antal sekunder sedan 1970-01-01. De användare som laddat en sida under de senaste tio minuterna räknas som inloggad. Även en lista med alla användare visas med deras namn som länkar till deras stad.

4.3 Databas

Databasen i BigCity är spelets minne. Där lagras all information om användarnas städer och aktioner mellan städerna. Även information som inte har direkt med användarnas städer att göra lagras i databasen, det kan vara information som talar om för spelmotorn vad den ska göra för uppdateringar eller meddelanden från spelledningen om förändringar i spelstrukturen.

4.3.1 MySQL

För att hantera en databas behövs ett databashanteringssystem. Till BigCity projektet så används MySQL för detta. MySQL är en databasserver med öppen källkod som använder sig av databasspråket SQL för att förändra, lägga till samt ta bort data i databasen. Öppen källkod

innebär att användare får använda och förändra programvaran utan att betala någonting. Källkoden får förändras så att den passar användarens egna mål. MySQL är ett relationsdatabashanteringssystem vilket innebär att databasen som MySQL jobbar mot är en relationsdatabas. En relationsdatabas lagrar data i många tabeller för att på så sätt öka hastighet och flexibilitet. MySQL är världens populäraste databasserver med över 5 miljoner servrar [1]. MySQL ingår i LAMP(Linux, Apache, MySQL, PHP) som är en programstack där alla delar i stacken har öppen källkod.

4.3.2 SQL

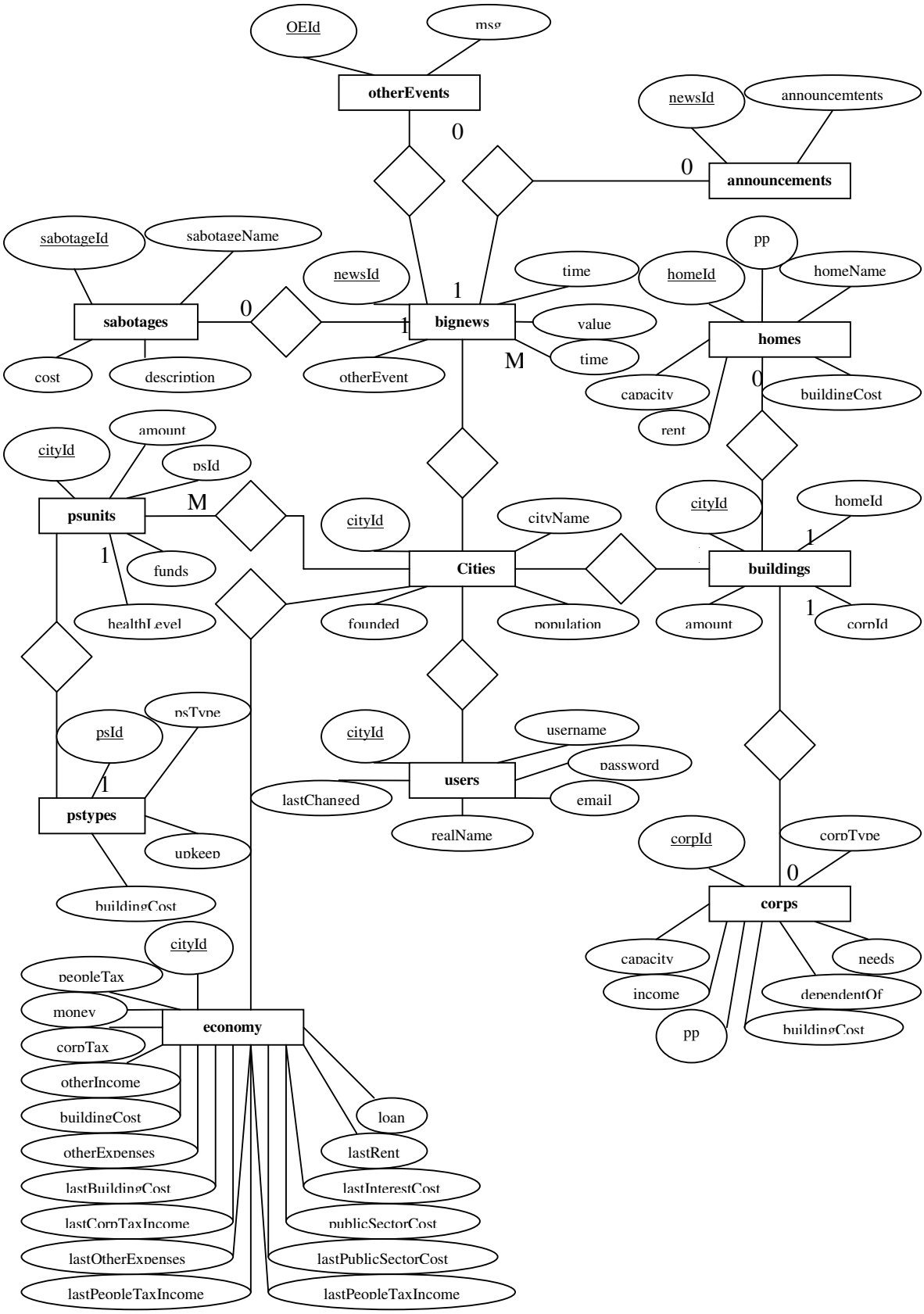
SQL står för ”Structured Query Language”, vilket översatt blir Strukturerat Frågespråk. SQL är ett relationsdatabasspråk Språket används av databashanteringssystem och bildar ett gränssnitt mellan databasen och databashanteringssystem. Databasspråket används av databashanteringssystemet för att utföra förändringar i databasen. Relationsmodellen som används i relationsdatabaser bygger i stort sett på tabeller som har rader och kolumner. Tabellerna lagras i databasen med bland annat definitionen på vad alla kolumner i databasen heter samt vad för typ av data som lagras där. SQL blev 1986 en ANSI standard och har sedan dess blivit en internationell standard av International Organization for Standardization(ISO) [1]. SQL används av många olika databashanteringssystem, bland annat MySQL som nämns i Kap 4.2.1.

4.3.3 BigCity-databasen

Relationsdatabasen som används i BigCity-projektet består utav 15 relationer, de flesta av dessa lagrar information om användarnas städer eller om interaktion mellan städerna. I Figur 4-9 visas ett ER-diagram över relationerna i databasen.

4.3.3.1 Stadsrelationer

Informationen om användarnas städer lagras i åtta relationer i databasen: users, cities, buildings, corps, homes, pstypes och psunits. Relationen users, se Tabell 4-1, lagrar information om användarens stadsid, användarnamn, lösenord, e-mail, riktiga namn samt när användaren senast uppdaterade en sida i spelet. Relationen används när någon försöker logga in i spelet, för att kontrollera om personen ifråga har en stad i spelet, och därmed rätt att logga in. Informationen om användarens riktiga namn och e-post lagras för att spelledningen ska kunna kontrollera om en användare har fler användarkonton än det tillåtna antalet. Fältet lastChanged används för att spelet ska kunna visa hur många spelare som för tillfället är



Figur 4-9 ER-dagram över databasen

aktiva samt för att göra så att en användare som inte laddat om någon sida i spelet under en viss tid tvingas logga in igen Att användaren tvingas logga in igen är en säkerhetsåtgärd ifall en användare lämnat spelet öppet på en offentlig dator.

Field	Type	Null	Default
cityId	int(11)	No	
username	varchar(11)	No	
password	varchar(11)	No	
email	varchar(30)	No	
realName	varchar(30)	No	
lastChanged	bigint(20)	No	0

Tabell 4-1 Relationen users

Relationen users har en koppling till relationen cities, där information om användarens stad finns lagrad. Kopplingen mellan de två relationerna sker genom fältet cityId som är primärnyckel, en primärnyckel är ett fält där värdet är unikt för varje tuple, i båda relationerna. I cities finns namnet på användarens stad, datumet då staden grundades samt storleken på populationen i staden. Relationen cities används ofta eftersom stadsnamnet visas för användaren på många ställen i spelet och populationen i staden behövs för att räkna ut bland annat antalet anställda i en stad, skatteintäkterna från privatperson och hur stor del av befolkningen som får del av en avdelning i den offentliga sektorn. Som synes i Tabell 4-2 har populationsfältet i cities ett defaultvärde 100. Så är fallet då 100 är startpopulation för städerna i BigCity.

Field	Type	Null	Default
cityId	int(11)	No	0
cityName	varchar(30)	No	
founded	date	No	0000-00-00
population	int(11)	No	100

Tabell 4-2 Relationen cities

I städer finns det byggnader, så är också fallet i BigCity. Byggnaderna i BigCity är av två olika bastyper, företag och bostäder. Databasen i BigCity använder sig av tre relationer för att lagra information om byggnader. För varje bastyp finns det en relation, dessa kopplas samman med cities-relationen genom relationen buildings. Relationerna corps och homes, Tabell 4-4 och Tabell 4-3, lagrar information om alla företags- och bostadstyper som finns. corps kan t.ex. innehålla information om företagstypen Stålfabrik, det kan vara information om t.ex. inkomster, stämningspoäng eller antalet personer som kan jobba i byggnaden.

Relationen buildings, Tabell 4-5, innehåller information om vilken stad som äger en viss byggnad, hur många av den sortens byggnad som staden äger och förstås information om vad för byggnad det är som staden äger.

Field	Type	Null	Default
homeId	int(11)	No	0
homeName	varchar(30)	No	
capacity	int(11)	No	0
rent	int(11)	No	0
pp	int(11)	No	0
buildingCost	int(11)	No	0

Tabell 4-3 Relationen homes

Field	Type	Null	Default
corpId	int(11)	No	0
corpType	varchar(30)	No	
capacity	int(11)	No	0
pp	int(11)	No	0
income	int(11)	No	0
buildingCost	int(11)	No	0
dependentOf	int(11)	No	-1
needs	float	No	0

Tabell 4-4 Relationen corps

Field	Type	Null	Default
cityId	int(11)	No	0
amount	int(11)	No	0
homeId	int(11)	Yes	*NULL*
corpId	int(11)	Yes	*NULL*

Tabell 4-5 Relationen buildings

Städerna i BigCity har en offentlig sektor som finns för att värna om städernas population. Den offentliga sektorn är viktig för att befolkningen ska hålla sig glada och stanna kvar i staden. Informationen om de olika avdelningarna i den offentliga sektorn finns lagrad i två relationer, psunits och pstypes. psunits liknar relationen buildings som nämns ovan, den lagrar information om vilken stad som äger avdelningen, vad det är för typ av avdelning, hur många enheter av avdelningen staden äger, health level(hur bra avdelningen tas om hand) och hur mycket resurser som har satts undan för att användas av avdelningen. Fältet psId i Tabell 4-6 används för att koppla samman relationen psunits med relationen pstypes.

Field	Type	Null	Default
cityId	int(11)	No	0
psId	int(11)	No	0
amount	int(11)	No	0
healthLevel	int(11)	No	0
funds	int(11)	No	50

Tabell 4-6 Relationen psunits

För att lagra beskrivningarna av de olika avdelningarna i den offentliga sektorn så används relationen pstypes, se Tabell 4-7. psId är primärnyckeln i relationen och länken mellan psunits och pstypes. psType är namnet på avdelningen som visas för användarna för att särskilja avdelningarna, upkeep är kostnaden för att underhålla avdelningen. Kostnaden för upkeep dras ifrån funds i psunits, om funds är tom så minskas health-level på avdelningen eftersom den inte underhålls. Fältet buildingCost i pstypes innehåller kostnaden för att bygga ut avdelningen. Kostnaden för att bygga ut avdelningarna i den offentliga sektorn kan alltså vara olika.

Field	Type	Null	Default
psId	int(11)	No	0
psType	varchar(30)	No	
upkeep	int(11)	No	0
buildingCost	int(11)	No	0

Tabell 4-7 Relationen pstypes

4.3.3.2 Interaktionsrelationer

För att BigCity ska kunna vara ett roligt och intressant onlinespel behöver användarna kunna interagera och tävla med varandra. Informationen om användarnas interaktion är av ett par olika slag, dels kan en användare sabotera för en annan användare, dels kan användare ha reklamkampanjer i varandras städer för att på så sätta locka över befolkning till sin egen stad. Information om dessa händelser behöver också komma de utsatta städerna till del så att kontraåtgärder kan göras. För att lagra information om ovannämnda saker finns det i BigCity-databasen fyra relationer, bignews, sabotages, announcement och otherEvents.

bignews är den relation som lagrar information om nyheter i BigCity. Nyheter kan vara att en stad blivit utsatt för sabotage eller något liknande. Det finns 3 olika sorters nyheter som var och en representeras av en relation (announcements, otherevents och sabotages). Relationen

bignews innehållet ett fält för varje sorts nyhet som finns. Dessa fält sätts till ett värde som motsvarar en specifik nyhet av den sorten som fältet i bignews representerar.

Field	Type	Null	Default
newsId	int(11)	No	
cityId	int(11)	No	0
responsibleId	int(11)	No	0
sabotageId	int(11)	No	-1
time	int(11)	No	0
value	int(11)	No	0
otherEvent	int(11)	No	-1
massEvent	int(11)	No	-1

Tabell 4-8 Relationen bignews

Fältet newsId i Tabell 4-8 är primärnyckel i relationen bignews, cityId används för att se vilken stad nyheten är till, responsibleId anger vilken stad som gjorde så att nyheten skapades. sabotageId används för att ange vad för sabotagetype det var som utfördes om någon, var det inget sabotage sätts värdet till -1. time är tidpunkten då nyheten skapades, lagrad i antal sekunder sedan 1970-01-01. value lagrar information som används vid visning av nyheten t.ex. antalet nedbrända företag vid en företagsbränning. otherEvent anger om det är en nyhet som inte passar in som en sabotagenyhet eller som massnyhet. Om nyheten inte är av typen otherevents sätts värdet på otherEvent till -1. massEvent fältet används för nyheter som är riktade av alla användare t.ex. spelinformation från spelledningen. Fältet massEvent sätts till -1 om nyheten inte är av typen announcements.

Relationen sabotages används i nyheterna när en användare utfört något sorts sabotage mot någon annan användare.

Field	Type	Null	Default
sabotageId	int(11)	No	0
sabotageName	Varchar(30)	No	
cost	int(11)	No	0
description	Text	No	

Tabell 4-9 Relationen sabotages

Tabell 4-9 visar hur relationen sabotages är uppbyggd. Fältet sabotageId i sabotages är kopplingen mellan sabotages och bignews. Fältet sabotageName lagrar sabotagenamnet, fältet cost innehåller kostnaden för att utföra sabotaget, description är en beskrivande text om sabotaget. Tabell 4-10 visar ett exempel på hur en tuple i relationen sabotages kan se ut.

sabotageId	sabotageName	cost	description
0	Bokföringsfiffel	50	Låt ett par av de mindre ärliga människorna i din stad fiffla med någon annan stads bokföring.

Tabell 4-10 Exempel på en tuple i relationen sabotages

announcements relationen används av spelledningen för att skicka ut nyheter till alla användare i BigCity. Fältet announcement, se Tabell 4-11, innehåller den information som spelledningen vill att användarna ska få del av medan fältet AId är kopplingen till relationen bignews. Samma värde som finns för tuplen i announcements relationen sätts i massEvent fältet i bignews.

Field	Type	Null	Default
AId	int(11)	No	
announcement	text	No	

Tabell 4-11 Relationen announcements

I BigCity skapas ett par nyheter som inte är av vare sig sabotage- eller announcementstyp. För att slippa skapa en ny typ för varje sådan nyhet finns det i databasen en relation otherEvents, Tabell 4-12. Nyheter som använder sig utav otherEvents relationen är t.ex. när en stad utför en reklamkampanj i en annan stad eller när en stad, som första stad, växer sig över en viss storlek.

Field	Type	Null	Default
OEId	int(11)	No	0
value	int(11)	Yes	-1
msg	text	Yes	*NULL*

Tabell 4-12 Relationen otherEvents

4.3.3.3 Övriga relationer

BigCity databasen innehåller ett par relationer utöver de ovannämnda. Dessa är relationer som inte används av spelfunktioner utan istället används för att användarna ska kunna kommunicera med varandra och med spelledningen.

4.4 Spelmotor

Olika spel har olika sätta att lagra information i sina databaser. De olika sätten att lagra information leder i sin tur till olika sätt att förändra och uppdatera denna information. Ett sätt att lagra användarens spelinformation är genom att hela tiden sätta aktuella värden i databasen. När förändringar inträffar uppdateras användarens spelinformation enligt förändringen och all information om användarens spelinformation finns därefter återigen uppdaterad och korrekt lagrad i databasen. Spel som använder detta sätt att lagra spelarinformationen kör oftast uppdatering på databasen vid vissa givna tillfällen. Detta medför att vid dessa tillfällen är det extra mycket uträkningar som behöver göras på servern vilket kan medföra att servern blir långsam eller till och med otillgänglig för användarna. Fördelarna med denna konstruktionslösning är att när användarens information ska hämtas för visning så behöver inga uträkningar göras på informationen utan den kan visas direkt. Det blir inte heller mycket uträkningar under tiden då spelet fortgår. Nackdelarna med detta system blir att när förändringar väl ska göras så utförs de på all användarinformation som finns vilket medför mycket tunga beräkningar.

Ett annat sätt att lagra användarinformationen är genom att låta den finnas lagrad i databasen men göra så att den inte behöver vara uppdaterad vid varje tidpunkt. Istället lagras det i databasen när användarinformation senast uppdaterades. Om sedan en ny förändring skall utföras så förändras först den användarinformation som finns i databasen så att den stämmer för den aktuella tidpunkten och sedan utförs den nya förändringen på den uppdaterade informationen. Detta sätt att lösa konstruktionen medför att viss information som hämtas och visas inte alltid är uppdaterad. Fördelen med att konstruera sitt spel på det här sättet är att det inte blir tunga beräkningar på vissa bestämda tidpunkter utan beräkningarna görs när det finns behov för dem. Nackdelen med lösningen är att spelet får arbeta med uträkningar direkt då de behövs av en användare, vilket medför att med många användare blir det mycket uppdateringar hela tiden för spelet. En annan nackdel är att information som inte är uppdaterad kan komma att visas för användaren.

De ovanstående sätten att lagra användarinformation leder till olika sätt att uppdatera informationen lagrad i spelens databaser. Spel som är inriktade mot att uppdatera sin användarinformation vid behov mer än på bestämda tidpunkter löser detta genom att uppdatera den specifika informationen med hjälp av de skriptspråk som spelen använder sig

av. Genom att använda samma skriptfiler som spelet är uppbyggt av så slipper programmerarna bry sig om externa sätt att uppdatera databasen. En nackdel med att låta alla beräkningar göras i skriptspråket är att de överlag är långsammare än språk som kräver kompilering. Detta beror på att skriptspråk använder sig utav en interpretator för att tolka koden när användaren begär att få se en sida. Interpretatorn tolkar koden och returnerar resultaten av koden i form av HTML. En interpretator har inte samma möjligheter att optimera som ett kompilerat språk har.

I BigCity utförs uppdateringar vid specifika tidpunkter. För att utföra dessa uppdateringar så har vi skapat ett fristående program, en spelmotor. Programmet är skrivet i C++ med ett API (Application Programmer Interface) mot databasen. API:t heter Mysql++ och är ett API skapat för att underlätta att skriva C++ program som ska jobba mot en Mysql databas. Här är ett exempel på hur en uppkoppling och fråga skapas och ställs mot en databas:

```
Connection con("DatabasNamn","DomänNamn","AnvändarNamn","Lösenord");
Query list_all_economy_query = con.query();
list_all_query << "SELECT * FROM economy";
Result res_economy = list_all_economy_query.store();
```

Spelmotorn läser vid uppstart in data från en textfil. Datan innehåller information till programmet hur och när det ska utföra uppdateringarna av databasen. Motorn hamnar sedan i en evighetsloop. I loopen kontrollerar motorn vad för dag det är och vad klockan är för tillfället, denna information jämförs sedan mot de klockslag då motorn ska uppdatera databasen. Om klockslagen och dagarna stämmer överens anropas olika funktioner för att utföra de uppdateringar som ska ske vid den specifika tidpunkten. Motorn har en funktion för varje typ av uppdatering som ska kunna utföras. Efter att motorn utfört eventuella uppdateringar så sover motorn i ett visst antal sekunder. Antalet sekunder motorn ska sova, innan nästa uppdateringskoll ska göras, läser motorn in från textfilen innan inträdet i loopen sker.

4.4.1 Populationsförändringar

För att utföra populationsförändringar i BigCity behöver spelmotorn hämta data från en stor del av databasen. De delar där data hämtas från är offentliga sektorn, ekonomin och byggnader. Populationen uppdateras vid specifika tidpunkter och förändringen i population beror på den stämning som råder bland befolkningen i staden. Stämningen i sin tur beror på vad för byggnader som finns i staden, vilka skattesatser som gäller och hur bra den offentliga sektorn är.

Motorn hämtar den data som behövs för uträkningarna och beräknar sedan utifrån detta fram en procentsats som förändrar den nuvarande populationen uppåt eller nedåt.

4.4.2 Förändringar i den offentliga sektorn

De förändringar som spelmotorn gör när offentliga sektorn ska uppdateras sker endast inom offentliga sektor-delen i databasen. Data om offentliga sektorn är uppdelad i två relationer. Den ena av relationerna innehåller information om de avdelningar som finns inom offentliga sektorn. I den andra relationen finns information om hur många enheter användarna har av varje avdelning. För att uppdatera offentliga sektorn loopar spelmotorn igenom alla tupler i relationen som lagrar data om antalet enheter. För varje tuple hämtar motorn data om den tuplens avdelningstyp. Sedan beräknar spelmotorn förändringarna på tuplen och uppdaterar den.

4.4.3 Ekonomiuppdateringar

Ekonomiuppdateringarna är omfattande och innebär mycket beräkningar. Det som sker är i stort sett att nuvarande inkomster och utgifter räknas ut och sättas som föregående veckas intäkter och kostnader. Beräkningarna blir mer omfattande på grund utav att inkomsterna från många byggnader är beroende på andra byggnader. Detta medför att motorn kan behöva kontrollera byggnader i flera led för att få fram om användaren ska få inkomster för en viss byggnad. Ekonomirelationen är den största av relationerna i databasen och har därför många fält att uppdatera.

5 Test

Detta kapitel behandlar de tester som vi utfört på motorn och php-koden i BigCity.

5.1 Php-kod

De tester vi gjort under och efter implementeringen av php-koden har endast varit manuellt utförda. Så är fallet eftersom de flesta php-sidorna lämnar ett visuellt resultat som behövs läsas för att kunna kontrollera om koden är korrekt. Även algoritmerna som används i php-koden har testats manuellt, detta har skett genom att värden har matats in till de olika php-sidorna.

5.2 Motorn

I motorn har tester gjorts under hela utvecklingen. De olika funktionerna i motorn, se kap 4.4, har utvecklats var och en för sig vilket medfört att de kunnat testas utförligt då all koncentration var riktad mot en funktion i taget. Testerna av motorns funktioner har varit manuella men ändå ganska omfattande då det är väldigt viktigt att motorn fungerar korrekt.

5.3 Prestandatester

För att testa hur väl motorn klarar av att hantera ett större antal användare, än de 15-20 användare som skapat konton under spelets utveckling, så skapades genom ett php-skript ytterligare 1000 konton. Dessa konton fick allting som en riktig användare får vid skapandet av ett konto. Det visade sig att motorn väl klarade av att hantera det större antalet användare. Vid en uppdatering med de riktiga 15-20 kontona så tog motorns uppdateringar inte längre än högst ett par sekunder. Med över 1000 användare tog samma uppdateringar 46 sekunder.

6 Problem

Detta kapitel påvisar och beskriver problem som kan uppstå vid skapandet och körning av online-spel. Alla problem som beskrivs är inte specifika för online-spel utan en del problem finns även i andra områden inom datavetenskap.

6.1 Hårdvara

Uppstår det problem i hårdvaran på en server som används i ett online-spel kan det skapa stor och ibland förödande skada på spelet. En hårdvarukrasch kan medföra att spelservern är nere i flera dagar tills ny hårdvara kan införskaffas. Beroende på vad för hårdvara som går sönder så kan fler problem uppstå.

Om hårddisken, där spelinformation lagras, kraschar måste informationen återhämtas från en backup, har dock ingen backup tagits på länge så kan viktig användarinformation gå förlorad. Annan hårdvara har inte de ytterligare problem som finns för en hårddiskkrasch, utan kan bytas ut och servern kan startas igen. Oavsett vad för hårdvara det är som kraschar så kan användarinformation, som skulle ha förändrats efter serverkraschen och innan servern återstartats, behöva förändras för att spelet ska fortgå på ett bra sätt. Här kommer ett exempel på ett fall där användarinformation i en backup skulle behöva uppdateras innan servern återstartar och användarna kan logga in igen.

I ett fotbollsspel kan användarna lägga ut sina spelare på auktion, en sluttid och datum sätts för auktionen. En användare, Johan, lägger ut en av sina spelare på auktion och slutdatum sätts ett par dagar framåt. Ett lågt bud läggs på spelaren men Johan är inte orolig för det är flera dagar kvar tills auktionen tar slut. Spelet tar efter det en backup och strax därefter så får spelservern problem och spelet är därefter inte nåbart. Efter ett par dagar är spelet återigen igång och nåbart. Dock så har tidpunkten för Johans auktion redan passerats och auktionen är därför slut. Johans duktiga spelare såldes för en småsumma eftersom det enbart fanns ett lågt bud på spelaren då backupen togs.

I ovanstående fall skulle tidpunkten för auktionens slut behöva förlängas innan spelservern åter görs nåbar online. Spelet skulle kunna fortgå utan att denna uppdatering görs men för att

ett spel ska vara rättvist och användarna glada så bör uppdateringen ske. De flesta online-spel är uppbyggda så att de kan hamna i en sådan situation som den ovan, detta är fallet eftersom i stort sett alla online-spel har uppdateringar som sker på vissa specifika tider eller som i fallet ovan har spelfunktioner som är tidsberoende.

6.2 Datorvirus

Datorvirus är ett problem som riskerar att drabba datorer som är uppkopplade till Internet om inte försiktighet råder. Problemet kan i stort sett avstyras med hjälp utav anti-virusprogram. En dedicerad online-spelserver, en server som enbart används som en online-spelserver, använder sig antagligen inte av anti-virusprogram på grund av prestandaskäl men bör ändå inte ha några problem med virus. Spelservern används aldrig till att ladda ner filer från Internet så det enda sättet för servern att bli infekterad är om infekterade filer installeras eller laddas upp till servern och detta kan speladministratörerna relativt lätt skydda sig mot genom att kontrollera filerna efter virus innan filerna skickas upp servern.

6.3 Webbläsarolikheter

I början av utvecklingen av spelet hade vi en del problem med webbläsarolikheter. Det var främst Opera under Linux som ställde till det då en del av de JavaScript som användes ej fungerade. Vi använde oss t.ex. av en dynamisk meny som expanderades vid klickning. När detta inte fungerade i Opera bestämde vi oss för att minimera mängden JavaScript för att spelet skulle gå att spela på de flesta moderna webbläsare. Det enda som JavaScript används för nu är små finesser som t.ex. att markören automatiskt fokuseras till inloggningsformuläret på inloggningssidan. Vi tycker dock inte att detta är något större problem då över 80 % av alla som surfar använder sig av Internet Explorer 5 eller 6 [3] vilka är de webbläsare vi använt oss av under utvecklingen av spelet.

7 Åsikter om spelet

För att få reda på vad användarna tycker om spelet har ett par frågor skickats ut till några slumpvist utvalda personer av våra testanvändare. Frågorna som ställdes var:

1. Vad tycker du om spelet BigCity i allmänhet?
2. Vad tycker du om utseendet och menyer?
3. Vad skulle du vilja förändrades i utseendet?
4. Vad tycker du om de spelfunktioner som finns?
   Är de bra/dåliga, för få/många? Vad skulle du vilja ändra på i spelets nuvarande funktioner?
5. Vad skulle du vilja lägga till för nya funktioner i spelet?
6. Vad har du för övriga åsikter om spelet?

Användare vic svarade såhär:

”1: Jag tycker att det är ett mycket bra spel, inte för att jag brukar spela så många spel.

2: Det är ett enkelt och snyggt utseende, lätt att hitta knapparna...

3: Inget just nu.

4: Det var inte några lätta frågor, framförallt inte till en amatör som jag. Jag saknar dock att det inte finns något slut på hur länge man kan hålla på? Jag tycker det var bra med sabotagegrejen, ganska roligt.

5: Inget just nu.

6: Jag tycker att ni har jobbat på bra pojkar o det är klar VG från mig, men misstänker att det inte är jag som får ge er VG:et.. Lycka till o jag återkommer om jag har något mer vettigt att skriva...Hmm”

Användare jemani svarade såhär:

”*Jag tycker spelet är kul när man väl har satt sig in i det och förstår vad det går ut på. Det kan kännas lite klurigt först när man inte riktigt fattar hur allt fungerar med den Offentliga sektorn och allt! Vet dock inte om jag fattat riktigt än =) Kanske tycker man skulle kunna fixa till utseendet så det ser lite mer "flashigt" och mer färgglatt. Menyerna tycker jag är bra men som sagt var tycker jag att det ser kanske lite "tamt" ut i själva outlooken. Det enda jag skulle vilja förändra med spelet är att man skulle kunna fylla i alla kategorier under Offentliga*

sektorn på samma gång så man slipper fylla i vilka man vill utöka och med hur mycket var för sig utan att det ska finnas ett alternativ som gör att alla kategorier fylls i samtidigt. Sedan skulle jag vilja se lite mer vad mina motspelare har för sig och hur de valt att bygga upp sin stad m.m. för att lättare kunna konkurrera."

Användare mufflon svarade såhär:
"1: Saknar ibland något att göra, det blir för mycket att bara logga in för att snabbt kolla offentliga sektorn och bygga villor och bondgårdar för alla pengar. Det är ett bra spel överlag, även om det finns för lite att pyssla med.

2: Bra menyer och som ett stort fan av blått kan jag ju inte klaga :)
Lättnavigerat menysystem, inget krångel där inte. Men frågan är ju hur det blir om nya saker läggs till…

3: Det finns egentligen inget. Det är enkelt och bra, möjligtvis kunde det finnas lite mer bilder, men samtidigt vet jag inte om det är värt att lägga ner alltför mycket tid på att göra några bilder nästan ingen bryr sig om i slutändan ändå. :) Men bilder ökar ändå helhetsintrycket.

4: De spelfunktioner som finns är ju bra, men de är på tok för få. Ge mig mer grejer att pyssla med!! Till exempel att man måste bygga kraftverk. När det gäller reklamkampanjer kanske det borde vara möjligt att göra det på mindre städer än en själv, dock med mindre effekt ju mindre de är i förhållande till sig själv. Sen så tror jag att ni har glömt en del av den offentliga sektorn, borde inte utbildning ligga där också? Skolor är väl kommunala antar vi :)

Det är även på tok för lätt att få en stor stad… jag hörde er nämna nåt om att göra allt svårare för en stor stad på något sätt, och jag kan väl säga att jag inte är emot det på något sätt, för som det är nu rasar invånarantalet iväg i en hisklig fart. Inflyttningshastigheten (och även utflyttningshastigheten!) måste begränsas mer!!!

Vilken skattesats som ska tillämpas borde också ändras… Det borde snarare vara en flytande än en fast gräns (19 % som det är nu), och ju fler invånare i staden desto lägre skattesats ska invånarna vilja ha… d.v.s. man måste sänka skattesatserna allteftersom man växer, annars stagnerar inflyttandet.

5: Angående kraftverk... Det borde finnas olika slags kraftverk att bygga där kraftverken ger olika mängder ström och föroreningar, och tillgången på el och den totala mängden föroreningar påverkar stämningen i staden (då borde också olika företag borde ge olika mängder föroreningar, t ex bilfabrik ger en del medan vissa inte ger några föroreningar alls, typ reklambyrå). Det borde också vara möjligt att bygga parker för att höja stämningen något. :)

Sen angående bondgårdsproblemet… Ni borde (läs måste) göra så att när det finns för många företag av en viss typ så avtar effekterna av dem (p.g.a. konkurrens, som ni skrev på reglersidan), annars kommer folk bara att bygga typ bondgårdar i all evighet. Alternativt kan stämningen som ges från en byggnadstyp minska om man har för många av den typen.

Dessutom (vilket kanske blir lite komplicerat att implementera) kanske det borde vara så att olika slags bostäder (och kanske företag) är bra för olika stora städer.. vad jag menar är att i en liten stad är det bäst med nästan bara villor, sen tar radhus över mer och mer och när staden blir riktigt stor ska höghus vara det mest givande att bygga (det blir mer verklighetstroget än att man bara bygger villor, vilken miljonstad har 99 % villor?). Vet inte hur ni skulle kunna göra detta, men det borde väl bara vara att göra antalet folk som kan bo i viss bostadstyp varierar med totala antalet invånare eller nåt. Ni får fundera på det. ;)

Och en sista grej...
Sopor! Det borde generas sopor, och man måste allokera ett visst stort område till soptipp.. Detta område måste sedan utökas allteftersom berget växer, vilket kostar pengar. Och om detta implementeras borde det också kanske finnas nåt slags sopförbränningsstation eller vad det heter som man kan bygga, som kan förbränna en viss mängd sopor på en viss tid för att minska sopberget och även generera lite el (om ni fixar kraftverks- och elbiten först). Båda sopberget och förbränningsstationerna kanske borde ge minskad stämning p.g.a. föroreningar.

6: Inget egentligen, bara att det finns för lite att pyssla med. :) Det är allt jag kan komma på just nu, kommer jag på mer så säger jag till eller skriver ett nytt meddelande. :)"

Användare eronile svarade såhär:

1: Det är frustrerande och beroendeframkallande, men det är väl ett gott betyg, med tanke på att det är så sådana här spel fungerar.

2: Jag tycker att det är snyggt, stilrent och enkelt. Får inte vara för mycket plotter, för då är det svårt att hitta rätt och man blir bara trött i huvudet.

3: Inget stort direkt. Kanske bara meddelandena, att det är mer tydligt vem man fått ifrån och så. Är rädd varje gång jag ska skriva att jag svarar till någon annan.

4: Tycker att de är bra, men skulle vilja veta mer vad det är som gör att folk vill flytta till staden, och vad som gör att man flyttar därifrån, så man får göra en avvägning. Det borde alltså vara dyrare med "feel good"-bygnaderna och billigare med de motsatta. Tycker nog att det kan finnas fler funktioner också. När man väl satt sig in i spelet finns det inte så mycket olika att välja på. Kunde alltså finnas fler än bygg/riv bostad och företag.

5: Jag skulle vilja ha in fler funktioner som kan ändras, men som inte behöver kosta något. Dessa funktioner kan alltså vara positiva i vissa sammanhang och negativa i andra. Detta behöver inte stå någonstans, utan man kan själv märka det genom att stämningen höjs eller inte. Tycker också att det skulle kunna finnas en värdemätare på stan där allt inkluderas. Så att det blir lite tävling mellan städerna. Nu kan en användare ha flest invånare och en annan mest pengar. Vad är det som säger vilken stad som är bäst? Tycker även att en spärr borde finnas som säger att man inte kan göra hur många reklamkampanjer som helst i en och samma stad.

6: Jag tror jag har sagt det mesta. Kommer jag på nåt mer håller jag er underrättade. Jag tycker att ni gör ett kanonjobb, och jag vet att det inte finns hur mycket tid som helst för att utveckla spelet. Lycka till i fortsättningen!

Åsikterna ovan är till stor del positiv, vilket är glädjande, men det finns också kritik att ta till sig. Överlag tycker användarna att det finns för lite att göra i spelet, detta är något som vi är väl medvetna om och har tänkt åtgärda, se kapitel 8. Utseendet på sidan är vi nöjda med och har därför inga nuvarande planer på att åtgärda klagomål inom detta område. Idén om att införa elbehov i byggnader är klart intressant och det finns nog chans att det kan komma att läggas till i spelet.

8 Framtida förändringar och tillägg

BigCity som det ser ut färdigt idag skiljer sig en del från de tankar och idéer som fanns när spelets skapande fortfarande låg i startgroparna. En del idéer har fått läggas på hyllan för tillfället och andra har fått ge vika för förändringar i spelets struktur. Detta kapitel tar upp förändringar samt funktionalitet vi skulle vilja lägga till i framtiden.

8.1 Handelsavtal

När BigCity skulle till att skapas fanns en idé om att användarna av spelet skulle kunna interagera med varandras städer genom handel. Ett sätt att utföra handel på var genom att användarna skulle kunna hyra ut en del av sina företag till en annan stad. Detta skulle ge extra intäkter till staden som hyr ut sina företag och staden som hyr in företag får fördelen av att slippa bygga dyra företag. Idén blev ännu mer intressant när förändringar gjordes i företagsstrukturen. Förändringen medförde att företag nu kunde behöva resurser från andra företag för att fungera t.ex. en stålfabrik behöver järnmalm från ett par malmgruvor för att fungera till 100%.

8.2 Kollektivtrafiksavtal

Genom att två städer sluter ett kollektivtrafiksavtal kan intäkterna i de två städerna öka något. Bussar köps in som sedan kör mellan de två städerna. En stor stad som har avtal med en liten stad får inte så mycket större intäkter medan den lilla staden kan få rätt så stora ökningar i intäkter. Startkostnaderna och underhåll av bussar krävs vilket kostar en del pengar. Denna idé fanns det inte tid för att implementera men är fortfarande väldigt intressant.

8.3 Katastrofer

En rolig idé som vi inte fick chans att implementera gick ut på att städerna i BigCity ibland skulle ha otur och råka ut för olika sorters katastrofer. Katastrofer skulle sedan motverkas av städernas offentliga sektor och byggnader. En sorts katastrof skulle t.ex. vara att bränder startar i en stad, dessa bränder skulle sedan släckas en ny avdelning, brandstationer, i den offentliga sektorn. Innan bränderna blivit släckta skulle byggnader ha brunnit ner och människor ha dött i bränderna. Om avdelningen hade dålig täckning och/eller dålig health-level skulle längre tid krävas för att få bränderna under kontroll.

8.4 Förändringar i nuvarande system

En del förändringar skulle vilja göras i en del av den nuvarande funktionaliteten i spelet. Det behöver inte vara stora saker utan kan vara små irritationsmoment

8.4.1 Användargränssnittet

Offentliga sektorn

När användaren ska bygga ut den offentliga sektorn eller tilldela någon avdelning i den offentliga sektorn pengar får användaren välja vilken avdelning som ska förändras. Vill användaren då förändra alla avdelningar blir det lite enformigt att göra samma sak för alla avdelningar. Därför skulle vi vilja lägga till ett val i drop-down menyerna där det går att välja att förändra alla avdelningar i den offentliga sektorn på en och samma gång.

Allmänt

På de sidor där användarna kan utföra olika operationer på sin egen och andras städer skulle det behöva finnas en länk till en sida där det finns information om operationerna användaren kan utföra.

8.4.2 Spelmotorn

Populationsförändringar

När en ny population, som är beroende på stämningen i staden, ska räknas ut tar motorn hänsyn till de satta skattesatserna i städerna. Är skattesatserna låga så ökar stämningspoängen i staden och är de höga så minskar stämningspoängen. Dock så är gränserna för när skattesatsen är låg respektive hög fast satta i det nuvarande systemet. En förändring borde göras så att gränserna inte längre är fast satta utan att en liten minskning i skattesatserna ger en liten ökning i stämningspoäng och en stor minskning ger en stor ökning i stämningspoäng och därmed antagligen en större population.

8.4.3 Databasen

Inga större förändringar behöver ske i databasen i det nuvarande systemet förutom att reflektera andra eventuella förändringar.

9 Resultat

I kapitel 7 Åsikter om spelet, tas ett par användares åsikter upp. Bland de positiva åsikterna kunde det utläsas att användarna överlag tycker att det är ett roligt spel som har ett bra gränssnitt med lättnavigerat menysystem. Den funktionalitet som finns i spelet är till belåtenhet. Det som användarna tycker är mindre bra med spelet är att det finns för lite saker att göra. Önskemål om fler funktioner som inte kostar pengar fanns. Även åsikter om att layouten var lite tråkig och färglös fanns.

Av de åsikter som frambringats till oss drar vi slutsatsen att användarna tycker det är kul att spela spelet, vilket är det viktigaste. Dock vill de ha mer att göra i spelet.

Den produkt vi utvecklat uppfyller till stor del de krav och mål som sattes upp innan examensarbetet påbörjades. Helt färdigutvecklat som spel är det dock inte. Även vi delar användarnas åsikter och tycker att det saknas en del funktionalitet för ett komplett online-spel. Det finns för få saker för användarna att förändra och syssla med. Av detta kan slutsatsen dras, att kravspecifikationen var för liten för ett komplett spel. Dock var det en bra kravspecifikation med tanke på den tid som fanns till att utveckla spelet. Även om spelet inte är helt komplett så är det en mycket bra grund att bygga vidare på och spelet bedöms ha potential för att kunna locka användare vid vidareutveckling.

Referenser

[1] *http://www.mysql.com 2004-05-17*

[2] *http://www.itl.nist.gov/div897/ctg/dm/sql_info.html 2004-05-18*

[3] http://www.w3schools.com/browsers/browsers_stats.asp 2004-05-19

[4] http://www.hattrick.org 2004-06-09

[5] http://www.stolpskott.net 2004-06-09

[6] http://www.managerzone.com 2004-06-09

[7] http://games.swirve.com/utopia 2004-06-09

[8] http://www.orkfia.net 2004-06-09

[9] http://www.kamikazegames.com/dominion 2004-06-09

[10] http://www.warriors2.com 2004-06-09

[11] http://vow.plit.dk 2004-06-09

A Spelmanual

Spelregler/Manual

- Befolkning
- BigNews
- Offentliga sektorn
- Intäkter
- Stadens skick
- Företag
- Bostäder
- Interaktion
    - Sabotage
    - Reklamkampanj

Befolkning
Förändras procentuellt från antal invånare mot ett "normalvärde" som styrs av faktorer som arbetstillfällen & den totala stämningspoängen.

BigNews
När något händer som berör din stad skapas det en löpsedel. Denna löpsedel kan ses på din stads sida. Saker som kommer med i BigNews just nu är sabotage, reklamkampanjer, rivning av bebodda hus & populationsförändringar över jämna tal.

Offentliga sektorn
Offentliga sektorn har en så kallad health-level(HL) som måste underhållas med pengar. Pengarna läggs i en buffert och "äts upp". Varje del i den offentliga sektorn kostar 1 iKr per 20 invånare den räcker till. När pengarna är slut sjunker HL t.ex. fem procentenheter. När sedan pengar sätts in i bufferten igen så stiger HL med t.ex. fyra procentenhet per dag. HL funkar så att om t.ex. polisens HL sjunker så minskar antalet människor i staden som varje station kan beskydda. Räcker inte avdelningarna i den offentliga sektorn till tillräckligt med folk så blir de sura och vill kanske inte bo i din stad.

Dessa delar ingår i den offentliga sektorn:
Infrastruktur
Polis

Sjukvård
Kultur
Lokaltrafik

Varje enhet räcker till ett visst antal människor och varje enhet kostar lika mycket.

Intäkter
Skatter: Det finns två skattesatser som användaren kan ändra. En för företag och en för privatpersoner. Ju lägre skattesats desto mer stämningspoäng, men lägre skatteintäkter då förstås. Företagen betalar skatt på deras intäkter och privatpersonerna på deras lön. Alla privatpersoner tjänar lika mycket, medan företagen har olika intäkter

Hyra: Staden får in hyra som räknas ut med befolkningsprocenttalet gånger varje typ av hus intäkter gånger antal hus av den typen.

Stadens skick
Stadens skick räknas ut genom att ta HL för varje del i den offentliga sektorn, gångra det med dess stadstäckning.

Företag
Vissa företag är beroende av andra företag för att kunna producera och tjäna pengar. En bilfabrik t.ex. behöver 3 stålfabriker som försörjer den med stål. Stålfabrikerna är i sin tur beroende av att det finns två malmgruvor per stålfabrik som kan leverera malm till dem. Företagen anställer olika många arbetare beroende på företagsområde. Stämningspoäng och intäkter skiljer sig också mellan olika områden. T.ex. så har ett tivoli inte så höga intäkter men höjer stämningspoängen och bilfabriken har hög inkomst, men sänker stämningspoängen.

Ett företag kostar aldrig pengar efter uppbyggnad. Har man byggt för många företag så får det bara mindre antal anställda och går således inte med mycket vinst.

Bostäder
En stad behöver bostäder för att göra plats för befolkningen. Bostäder inbringar hyra och stämningspoäng beroende på bostadstyp. De bostäder som står tomma inbringar förstås inga hyresintäkter.

Interaktion mellan städer
I spelet kan spelarna interagera med varandra genom dessa operationer.
*Sabotage
*Reklamkampanj

Sabotage
En stad kan sabotera för en annan stad. Chansen att sabotaget misslyckas och att det upptäcks varifrån sabotaget beordrades beror på hur bra polisen är i staden där sabotaget utförs. Ju sämre

polis du själv har desto lättare är det att få tag i bra fifflare som ökar chanserna för att ett sabotage lyckas. Vid ett totalt misslyckande så får sabotören betala böter för brottet. Operation kan också lyckas men sabotören kan dock bli känd dock utan tillräckliga bevis för en fällning i rätten.
Olika sabotage som kan genomföras är:

Bränna fabriker: Ett visst antal fabriker bränns ner under mystiska omständigheter. Ingen dör dock i branden, ej heller skadas några djur.

Bokföringsfiffel: Ett misstag i bokföringen för den drabbade staden gör att en stor summa pengar försvinner. Staden som utför operationen får dock ej pengarna.

Förstöra infrastruktur: Operationen går ut på att förstöra elledningar, vägar och annan infrastruktur i den andra staden.

Förstöra lokaltrafik: Liknar ovanstående operation. Sabotage mot motståndarstaden genomförs så att lokaltrafiken i staden försämras. Kan ske genom punkteringar, klotter och annan illegal verksamhet.

Reklamkampanj
Om du inte är nöjd med befolkningsantalet i din stad kan reklamkampanjer utföras i andra städer för att på så sätt locka över dess befolkning till din egen stad. Staden som utför kampanjen får en bonus som beror på antalet reklamföretag som finns i staden. Tänk på att ha bra förutsättningar och plats för din nya befolkning, för får de inte plats så stannar de kvar i staden de bor i.

B Ursprunglig specifikation

Befolkningsantal

Förändras procentuellt från antal invånare mot ett ”normalvärde” som styrs av faktorer som arbetstillfällen och den totala stämningspoängen.

Löpsedel

Varje dag skapas en löpsedel där det står en nyhet som ger en indikation på läget i staden.
Ex: Kvinna rånad på Sveagatan! Rånaren skottskadad och förkyld.
Detta tyder alltså på hög brottslighet.

Offentliga sektorn

Offentliga sektorn har en så kallad health-level (HL) som måste underhållas med pengar. Pengarna läggs i en buffert och ”äts upp”. När pengarna är slut sjunker HL t.ex. fem procentenheter. När sedan pengar sätts in i bufferten igen så stiger HL med t.ex. en procentenhet per dag. HL funkar så att om t.ex. polisens HL sjunker så minskar antalet människor i staden som varje station kan beskydda.

Dessa ingår i den offentliga sektorn:
Infrastruktur
Polis
Sjukvård
Parker, museum, bibliotek
Lokaltrafik

Infrastruktur och lokaltrafik kostar en viss summa för antalet människor den ska nå ut till, och det är en grundkostnad för varje utbyggnad. Hos de andra så räcker varje enhet till ett visst antal människor och varje enhet kostar lika mycket.

Intäkter

Skatter: Det finns två skattesatser som användaren kan ändra. En för företag och en för privatpersoner. Ju lägre skattesats desto mer stämningspoäng, men lägre skatteintäkter då förstås. Företagen betalar skatt på deras intäkter och privatpersonerna på deras lön. Alla privatpersoner tjänar lika mycket, medan företagen har olika intäkter

Hyra: Staden får in hyra som räknas ut med befolkningsprocent-talet gånger varje typ av hus intäkter gånger antal hus av den typen.

Stadens skick

Stadens skick räknas ut genom att ta HL för varje del i den offentliga sektorn, gångra det med dess stadstäckning.

Företag

Man bör inte ha för många företag av en typ. Då minskar intäkterna då konkurrensen blir för stor. Detta antal är beroende av stadens storlek, innevånare. Företagen anställer olika många arbetare beroende på företagsområde. Stämningspoäng och intäkter skiljer sig också mellan olika områden. T.ex. så har ett tivoli inte så höga intäkter men däremot ett högt stämningspoäng.

Bostäder

En stad behöver bostäder för att göra plats för befolkningen. Bostäder inbringar lite hyra och lite stämningspoäng.

Interaktion mellan städerna

I spelet kan spelarna interagera med varandra genom dessa operationer.

*Sabotage

*Reklamkampanj

*Handelsavtal

Se respektive operations info

Sabotage

En stad kan sabotera för en annan stad. Chansen att sabotaget misslyckas och att det upptäcks varifrån sabotaget beordrades beror på hur bra polisen är i staden där sabotaget utförs. Vid ett

totalt misslyckande så får sabotören betala böter för brottet. Operation kan också lyckas men sabotören kan dock bli känd dock utan tillräckliga bevis för en fällning i rätten.

Olika sabotage som kan genomföras är:
Bränna fabriker: Ett visst antal fabriker bränns ner under mystiska omständigheter. Ingen dör dock i branden, ej heller skadas några djur.
Bokföringsfiffel: Ett misstag i bokföringen för den drabbade staden gör att en stor summa pengar försvinner. Staden som utför operationen får dock ej pengarna.
Förstöra infrastruktur: Operationen går ut på att förstöra elledningar., vägar och annan infrastruktur i den andra staden.
Förstöra lokaltrafik: Liknar ovanstående operation. Sabotage mot motståndarstaden genomförs så att lokaltrafiken i staden försämras. Kan ske genom punkteringar, klotter och annan illegal verksamhet.

Reklamkampanj
Om befolkningsantalet i staden är lågt kan reklamkampanjer utföras i andra städer, för att på så sätt locka över dess befolkning till sin egen stad. Stämningen i städerna jämförs för att se om kampanjen lyckades. Staden som utför kampanjen får en bonus som beror på antalet reklamföretag som finns i staden. (Vi avvaktar med att införa iden med att kunna locka över företag m.h.a. reklamkampanjer.)

Handelsavtal
Två städer kan sluta ett handelsavtal som sträcker sig ett bigcity år framåt. Ett Handelsavtal går ut på att en stads företag, inom ett visst företagsområde, utvidgar sin affärsverksamhet till den andra staden i handelsavtalet. Om den andra staden har liten verksamhet så får företagen en större marknad och intäkterna för företaget ökar då. Skatten på denna ökning delas mellan de två städerna enligt ett visst förhållande. En stad kan ej erbjudas ett handelsavtal inom ett företagsområde om staden redan har 50% eller mer av den möjliga marknaden.

C Kod

Se medföljande Cd-skiva.